1. Builder Design Pattern
2. Source code:

**package** Question\_1;  
  
**public class** PizzaStore {  
 String **storeName**;  
 Pizza **pizza**;  
  
 **public** PizzaStore(String storeName, Pizza pizza) {  
 **this**.**storeName** = storeName;  
 **this**.**pizza** = pizza;  
 }  
  
 **public void** eat(){  
 System.***out***.println(**"Welcome to "** + **storeName**);  
 **this**.**pizza**.info();  
 }  
}

**package** Question\_1;  
  
**public abstract class** Toppings {  
 **boolean pepperoni**;  
 **boolean saussage**;  
 **boolean mushrooms**;  
 **boolean bacon**;  
 **boolean onions**;  
 **boolean extraCheese**;  
 **boolean peppers**;  
 **boolean chicken**;  
 **boolean olive**;  
 **boolean spinach**;  
 **boolean tomatoAndBasil**;  
 **boolean beef**;  
 **boolean ham**;  
 **boolean pesto**;  
 **boolean spicyPork**;  
 **boolean hamAndPineapple**;  
}

**package** Question\_1;  
  
  
**import** java.util.HashMap;  
  
**public class** Pizza **extends** Toppings {  
  
 **private int size**;  
 **private** HashMap<String, Boolean> **orderedToppings** = **new** HashMap<String, Boolean>();  
 **public** Pizza(PizzaBuilder builder){  
 **this**.**size** = builder.**size**;  
 **this**.**pepperoni** = builder.**pepperoni**;  
 **this**.**saussage** = builder.**saussage**;  
 **this**.**mushrooms** = builder.**mushrooms**;  
 **this**.**bacon** = builder.**bacon**;  
 **this**.**onions** = builder.**onions**;  
 **this**.**extraCheese** = builder.**extraCheese**;  
 **this**.**peppers** = builder.**peppers**;  
 **this**.**chicken** = builder.**chicken**;  
 **this**.**olive** = builder.**olive**;  
 **this**.**spinach** = builder.**spinach**;  
 **this**.**tomatoAndBasil** = builder.**tomatoAndBasil**;  
 **this**.**beef** = builder.**beef**;  
 **this**.**ham** = builder.**ham**;  
 **this**.**pesto** = builder.**pesto**;  
 **this**.**spicyPork** = builder.**spicyPork**;  
 **this**.**hamAndPineapple** = builder.**hamAndPineapple**;  
 **this**.**orderedToppings** = builder.**orderedToppings**;  
 }  
  
 **public void** info(){  
 HashMap<Integer, String> pizzaSize = **new** HashMap<>();  
 pizzaSize.put(3, **"small"**);  
 pizzaSize.put(6, **"medium"**);  
 pizzaSize.put(9, **"large"**);  
 System.***out***.printf(**"You have ordered a %s pizza with %d toppings: "**,  
 pizzaSize.get(**size**), **orderedToppings**.size());  
 **for** (String topping : **orderedToppings**.keySet()) {  
 **if** (**orderedToppings**.get(topping)) {  
 System.***out***.printf(**"%s, "**,topping);  
 }  
 }  
 System.***out***.println(**"\nEnjoy your pizza!\n"**);  
 }  
  
 **public static class** PizzaBuilder **extends** Toppings{  
 **private int size**;  
 **private** HashMap<String, Boolean> **orderedToppings** = **new** HashMap<String, Boolean>();  
  
 **public** PizzaBuilder(**int** size){  
 **this**.**size** = size;  
 }  
  
 **public** PizzaBuilder addPepperoni(**boolean** pepperoni){  
 **this**.**pepperoni** = pepperoni;  
 **orderedToppings**.put(**"pepperoni"**, pepperoni);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addSaussage(**boolean** saussage){  
 **this**.**saussage** = saussage;  
 **orderedToppings**.put(**"saussage"**, saussage);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addBacon(**boolean** bacon) {  
 **this**.**bacon** = bacon;  
 **orderedToppings**.put(**"bacon"**, bacon);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addOnions(**boolean** onions) {  
 **this**.**onions** = onions;  
 **orderedToppings**.put(**"onions"**, onions);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addExtraCheese(**boolean** extraCheese) {  
 **this**.**extraCheese** = extraCheese;  
 **orderedToppings**.put(**"extra cheese"**, extraCheese);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addPeppers(**boolean** peppers) {  
 **this**.**peppers** = peppers;  
 **orderedToppings**.put(**"peppers"**, peppers);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addChicken(**boolean** chicken) {  
 **this**.**chicken** = chicken;  
 **orderedToppings**.put(**"chicken"**, chicken);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addOlive(**boolean** olive) {  
 **this**.**olive** = olive;  
 **orderedToppings**.put(**"olives"**, olive);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addSpinach(**boolean** spinach) {  
 **this**.**spinach** = spinach;  
 **orderedToppings**.put(**"spinach"**, spinach);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addTomatoAndBasil(**boolean** tomatoAndBasil) {  
 **this**.**tomatoAndBasil** = tomatoAndBasil;  
 **orderedToppings**.put(**"tomato and basil"**, tomatoAndBasil);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addBeef(**boolean** beef) {  
 **this**.**beef** = beef;  
 **orderedToppings**.put(**"beef"**, beef);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addMushrooms(**boolean** mushrooms) {  
 **this**.**mushrooms** = mushrooms;  
 **orderedToppings**.put(**"mushrooms"**, mushrooms);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addHam(**boolean** ham) {  
 **this**.**ham** = ham;  
 **orderedToppings**.put(**"ham"**, ham);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addPesto(**boolean** pesto) {  
 **this**.**pesto** = pesto;  
 **orderedToppings**.put(**"pesto"**, pesto);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addSpicyPork(**boolean** spicyPork) {  
 **this**.**spicyPork** = spicyPork;  
 **orderedToppings**.put(**"spicy pork"**, spicyPork);  
 **return this**;  
 }  
  
 **public** PizzaBuilder addHamAndPineapple(**boolean** hamAndPineapple) {  
 **this**.**hamAndPineapple** = hamAndPineapple;  
 **orderedToppings**.put(**"ham and pineapple"**, hamAndPineapple);  
 **return this**;  
 }  
  
 **public** Pizza build() {  
 **return new** Pizza(**this**);  
 }  
  
 }  
}

**package** Question\_1;  
  
**public class** PizzaDriver {  
  
 **public static void** builderTest() {  
 *createPizzaHut*();  
 *createLittleCeasarsPizza*();  
 *createDominoPizza*();  
 }  
  
 **public static void** createDominoPizza(){  
 PizzaStore smallDomino = **new** PizzaStore(**"Domino"**,  
 **new** Pizza.PizzaBuilder(3).  
 addChicken(**true**).  
 build());  
 smallDomino.eat();  
  
 PizzaStore largeDomino = **new** PizzaStore(**"Domino"**,  
 **new** Pizza.PizzaBuilder(9).  
 addSpicyPork(**true**).  
 addHamAndPineapple(**true**).  
 addMushrooms(**true**).build());  
 largeDomino.eat();  
 }  
  
 **public static void** createLittleCeasarsPizza(){  
 PizzaStore smallLittleCeasars = **new** PizzaStore(**"Little Caesars"**,  
 **new** Pizza.PizzaBuilder(6).  
 addHam(**true**).  
 addHamAndPineapple(**true**).  
 addPeppers(**true**).  
 addTomatoAndBasil(**true**).  
 addMushrooms(**true**).  
 addSaussage(**true**).build());  
 smallLittleCeasars.eat();  
  
 PizzaStore largeLittleCeasars = **new** PizzaStore(**"Little Caesars"**,  
 **new** Pizza.PizzaBuilder(9).  
 addBeef(**true**).  
 addMushrooms(**true**).  
 addPeppers(**true**).  
 addExtraCheese(**true**).  
 addOlive(**true**).  
 addOnions(**true**).  
 addPepperoni(**true**).  
 addPesto(**true**).build());  
 largeLittleCeasars.eat();  
 }  
  
 **public static void** createPizzaHut(){  
 PizzaStore smallPizzaHut = **new** PizzaStore(**"Pizza Hut"**,  
 **new** Pizza.PizzaBuilder(3).  
 addPepperoni(**true**).  
 addSpinach(**true**).build());  
 smallPizzaHut.eat();  
  
 PizzaStore largePizzaHut = **new** PizzaStore(**"Pizza Hut"**,  
 **new** Pizza.PizzaBuilder(9).  
 addBacon(**true**).  
 addMushrooms(**true**).  
 addOnions(**true**).build());  
 largePizzaHut.eat();  
 }  
  
}

1. Outputs

Graphical user interface, text, application, email

Description automatically generated

1. Factory, Singleton Design Pattern
2. Source code

**package** Question\_2;  
  
**public class** FactoryDriver {  
 **public static void** factoryTest() {  
 *showCarFactory*();  
 *showAirplaneFactory*();  
 *showBoatFactory*();  
 }  
  
  
 **public static void** showBoatFactory(){  
 BoatFactory seaRayFactory = Factory.*createBoatFactory*(**"Sea Ray"**);  
 seaRayFactory.build(**"Sundeck"**, 2020, 2);  
 seaRayFactory.repair();  
  
 BoatFactory mastercraftFactory = Factory.*createBoatFactory*(**"Mastercraft"**);  
 mastercraftFactory.build(**"X-14"**, 2021, 2);  
 mastercraftFactory.restore();  
  
 BoatFactory bertramFactory = Factory.*createBoatFactory*(**"Bertram"**);  
 bertramFactory.build(**"Moppie"**, 2022,4);  
  
 *//TEST SINGLETON* BoatFactory anotherBertramFactory = Factory.*createBoatFactory*(**"Bertram"**);  
 anotherBertramFactory.build(**"Moppie"**, 2023, 2);  
  
 }  
 **public static void** showAirplaneFactory(){  
 AirplaneFactory boeingFactory = Factory.*createAirplaneFactory*(**"Boeing"**);  
 boeingFactory.build(**"737"**, 2020, 2);  
  
 AirplaneFactory airbusFactory = Factory.*createAirplaneFactory*(**"Airbus"**);  
 airbusFactory.build(**"A320"**, 2021, 2);  
  
 AirplaneFactory embraerFactory = Factory.*createAirplaneFactory*(**"Embraer"**);  
 embraerFactory.build(**"E190"**, 2022,4);  
  
 *//TEST SINGLETON* AirplaneFactory anotherEmbraerFactory = Factory.*createAirplaneFactory*(**"Embraer"**);  
 anotherEmbraerFactory.build(**"E195"**, 2023, 2);  
 }  
  
 **public static void** showCarFactory(){  
 CarFactory hondaCarFactory = Factory.*createCarFactory*(**"Honda"**);  
 hondaCarFactory.build(**"truck"**, 2023);  
  
 CarFactory porscheCarFactory = Factory.*createCarFactory*(**"Porsche"**);  
 porscheCarFactory.build(**"sport"**, 2020);  
  
 CarFactory teslaCarFactory = Factory.*createCarFactory*(**"Tesla"**);  
 teslaCarFactory.build(**"sedan"**, 2023);  
  
 *// TEST SINGLETON* CarFactory anotherTeslaCarFactory = Factory.*createCarFactory*(**"Tesla"**);  
 anotherTeslaCarFactory.build(**"SUV"**, 2022);  
  
 }  
}

**package** Question\_2;  
  
**public abstract class** Factory {  
 **public static** CarFactory createCarFactory(String type) {  
 **switch** (type) {  
 **case "Honda"**:  
 **return** HondaCarFactory.*getInstance*();  
 **case "Porsche"**:  
 **return** PorscheCarFactory.*getInstance*();  
 **case "Tesla"**:  
 **return** TeslaCarFactory.*getInstance*();  
 **default**:  
 **throw new** IllegalArgumentException(**"Invalid car type: "** + type);  
 }  
 }  
  
  
 **public static** AirplaneFactory createAirplaneFactory(String type){  
 **switch** (type) {  
 **case "Boeing"**:  
 **return** BoeingFactory.*getInstance*();  
 **case "Airbus"**:  
 **return** AirbusFactory.*getInstance*();  
 **case "Embraer"**:  
 **return** EmbraerFactory.*getInstance*();  
 **default**:  
 **throw new** IllegalArgumentException(**"Invalid airplane type: "** + type);  
 }  
 }  
  
 **public static** BoatFactory createBoatFactory(String type){  
 **switch** (type) {  
 **case "Sea Ray"**:  
 **return** SeaRayFactory.*getInstance*();  
 **case "Mastercraft"**:  
 **return** MastercraftFactory.*getInstance*();  
 **case "Bertram"**:  
 **return** BertramFactory.*getInstance*();  
 **default**:  
 **throw new** IllegalArgumentException(**"Invalid boat type: "** + type);  
 }  
 }  
}

**package** Question\_2;  
  
**public interface** Buildable {  
 **void** build(String type, **int** year);  
}

**package** Question\_2;  
  
**public interface** Repairable {  
 **void** repair();  
}

**package** Question\_2;  
  
**public interface** Restorable {  
 **void** restore();  
}

**package** Question\_2;  
  
**abstract class** CarFactory **implements** Buildable, Repairable, Restorable{  
 String **carType**;  
 **int year**;  
  
 **public** String toString(){  
 **return "Type: "** + **carType** + **" | Year: "** + **year** +**"\n"**;  
 }  
}

**package** Question\_2;  
  
**abstract class** AirplaneFactory **implements** Buildable, Repairable, Restorable{  
 String **airplaneType**;  
 **int engineCount**;  
 **int year**;  
  
 **public void** build(String airplaneType, **int** year){  
 **this**.**airplaneType** = airplaneType;  
 **this**.**year** = year;  
 }  
  
 **public** String toString(){  
 **return** String.*format*(**"Type: %s | Engine Count: %d | Year: %d%n"**, **airplaneType**, **engineCount**, **year**);  
 }  
  
 **public abstract void** build(String airplaneType, **int** year, **int** engineCount);  
}

**package** Question\_2;  
  
**abstract class** BoatFactory **implements** Buildable, Repairable, Restorable {  
 String **boatType**;  
 **int length**;  
 **int year**;  
  
 **public abstract void** build(String boatType, **int** year, **int** length);  
  
 **public void** build(String boatType, **int** year){  
 **this**.**boatType** = boatType;  
 **this**.**year** = year;  
 }  
  
 @Override  
 **public** String toString() {  
 **return** String.*format*(**"Type: %s | Length: %d | Year: %d%n"**, **boatType**, **length**, **year**);  
 }  
}

**package** Question\_2;  
  
**public class** HondaCarFactory **extends** CarFactory{  
 **static final** String ***CAR\_NAME*** = **"Honda Car"**;  
 **private static** HondaCarFactory *hondaFactotry*;  
  
 **private** HondaCarFactory() {  
 **super**();  
 }  
  
 **public static** HondaCarFactory getInstance() {  
 **if** (*hondaFactotry* == **null**) {  
 *hondaFactotry* = **new** HondaCarFactory();  
 } **else** {  
 System.***out***.printf(**"Warning! Only 1 %s Factory can be created. "** +  
 **"The existing factory will be used to build the new car%n"**, ***CAR\_NAME***);  
 }  
 **return** *hondaFactotry*;  
 }  
  
 **public void** build(String carType, **int** year){  
 **this**.**carType** = carType;  
 **this**.**year** = year;  
 showInfoCar();  
 }  
  
 **public void** repair(){  
 System.***out***.printf(**"Repairing %s%n"**, ***CAR\_NAME***);  
 }  
  
 **public void** restore(){  
 System.***out***.printf(**"Restoring %s%n"**, ***CAR\_NAME***);  
 }  
  
 **public void** showInfoCar() {  
 System.***out***.printf(**"The %s Factory have built a car with following details:%n"**,***CAR\_NAME***);  
 System.***out***.println(**super**.toString());  
 }  
}

**package** Question\_2;  
  
**public class** TeslaCarFactory **extends** CarFactory{  
 **static final** String ***CAR\_NAME*** = **"Tesla"**;  
 **private static** TeslaCarFactory *teslaFactory*;  
  
 **private** TeslaCarFactory() {  
 **super**();  
 }  
  
 **public static** TeslaCarFactory getInstance() {  
 **if** (*teslaFactory* == **null**) {  
 *teslaFactory* = **new** TeslaCarFactory();  
 }**else** {  
 System.***out***.printf(**"Warning! Only 1 %s Factory can be created. "** +  
 **"The existing factory will be used to build the new car%n"**, ***CAR\_NAME***);  
 }  
 **return** *teslaFactory*;  
 }  
  
 **public void** build(String carType, **int** year){  
 **this**.**carType** = carType;  
 **this**.**year** = year;  
 showInfoCar();  
 }  
  
 **public void** repair(){  
 System.***out***.printf(**"Repairing %s%n"**, ***CAR\_NAME***);  
 }  
  
 **public void** restore(){  
 System.***out***.printf(**"Restoring %s%n"**, ***CAR\_NAME***);  
 }  
  
 **public void** showInfoCar() {  
 System.***out***.printf(**"The %s Factory have built a car with following details:%n"**, ***CAR\_NAME***);  
 System.***out***.println(**super**.toString());  
 }  
}

**package** Question\_2;  
  
**public class** PorscheCarFactory **extends** CarFactory{  
 **static final** String ***CAR\_NAME*** = **"Porsche Car"**;  
 **private static** PorscheCarFactory *porscheFactory*;  
  
 **private** PorscheCarFactory() {  
 **super**();  
 }  
  
 **public static** PorscheCarFactory getInstance() {  
 **if** (*porscheFactory* == **null**) {  
 *porscheFactory* = **new** PorscheCarFactory();  
 }**else** {  
 System.***out***.printf(**"Warning! Only 1 %s Factory can be created. "** +  
 **"The existing factory will be used to build the new car%n"**, ***CAR\_NAME***);  
 }  
 **return** *porscheFactory*;  
 }  
  
 **public void** build(String carType, **int** year){  
 **this**.**carType** = carType;  
 **this**.**year** = year;  
 showInfoCar();  
 }  
  
 **public void** repair(){  
 System.***out***.printf(**"Repairing %s%n"**, ***CAR\_NAME***);  
 }  
  
 **public void** restore(){  
 System.***out***.printf(**"Restoring %s%n"**, ***CAR\_NAME***);  
 }  
  
 **public void** showInfoCar() {  
 System.***out***.printf(**"The %s Factory have built a car with following details:%n"**, ***CAR\_NAME***);  
 System.***out***.println(**super**.toString());  
 }  
}

**package** Question\_2;  
  
**public class** AirbusFactory **extends** AirplaneFactory {  
 **static final** String ***AIRPLANE\_NAME*** = **"Airbus Airplane"**;  
 **private static** AirbusFactory *airbusFactory*;  
  
 **private** AirbusFactory() {  
 **super**();  
 }  
  
 **public static** AirbusFactory getInstance() {  
 **if** (*airbusFactory* == **null**) {  
 *airbusFactory* = **new** AirbusFactory();  
 } **else** {  
 System.***out***.printf(**"Warning! Only 1 %s Factory can be created. "** +  
 **"The existing factory will be used to build the new airplane%n"**, ***AIRPLANE\_NAME***);  
 }  
 **return** *airbusFactory*;  
 }  
  
 @Override  
 **public void** build(String airplaneType, **int** year, **int** engineCount){  
 **this**.**airplaneType** = airplaneType;  
 **this**.**year** = year;  
 **this**.**engineCount** = engineCount;  
 showInfo();  
 }  
  
 **public void** repair(){  
 System.***out***.printf(**"Repairing %s%n"**, ***AIRPLANE\_NAME***);  
 }  
  
 **public void** restore(){  
 System.***out***.printf(**"Restoring %s%n"**, ***AIRPLANE\_NAME***);  
 }  
  
 **public void** showInfo() {  
 System.***out***.printf(**"The %s Factory have built an airplane with following details:%n"**, ***AIRPLANE\_NAME***);  
 System.***out***.println(**super**.toString());  
 }  
}

**package** Question\_2;  
  
**public class** BoeingFactory **extends** AirplaneFactory {  
 **static final** String ***AIRPLANE\_NAME*** = **"Boeing Airplane"**;  
 **private static** BoeingFactory *boeingFactory*;  
  
 **private** BoeingFactory() {  
 **super**();  
 }  
  
 **public static** BoeingFactory getInstance() {  
 **if** (*boeingFactory* == **null**) {  
 *boeingFactory* = **new** BoeingFactory();  
 } **else** {  
 System.***out***.printf(**"Warning! Only 1 %s Factory can be created. "** +  
 **"The existing factory will be used to build the new airplane%n"**, ***AIRPLANE\_NAME***);  
 }  
 **return** *boeingFactory*;  
 }  
  
 @Override  
 **public void** build(String airplaneType, **int** year, **int** engineCount){  
 **this**.**airplaneType** = airplaneType;  
 **this**.**year** = year;  
 **this**.**engineCount** = engineCount;  
 showInfo();  
 }  
  
 **public void** repair(){  
 System.***out***.printf(**"Repairing %s%n"**, ***AIRPLANE\_NAME***);  
 }  
  
 **public void** restore(){  
 System.***out***.printf(**"Restoring %s%n"**, ***AIRPLANE\_NAME***);  
 }  
  
 **public void** showInfo() {  
 System.***out***.printf(**"The %s Factory have built an airplane with following details:%n"**, ***AIRPLANE\_NAME***);  
 System.***out***.println(**super**.toString());  
 }  
}

**package** Question\_2;  
  
**public class** EmbraerFactory **extends** AirplaneFactory{  
 **static final** String ***AIRPLANE\_NAME*** = **"Embraer Airplane"**;  
 **private static** EmbraerFactory *embraerFactory*;  
  
 **private** EmbraerFactory() {  
 **super**();  
 }  
  
 **public static** EmbraerFactory getInstance() {  
 **if** (*embraerFactory* == **null**) {  
 *embraerFactory* = **new** EmbraerFactory();  
 } **else** {  
 System.***out***.printf(**"Warning! Only 1 %s Factory can be created. "** +  
 **"The existing factory will be used to build the new airplane%n"**, ***AIRPLANE\_NAME***);  
 }  
 **return** *embraerFactory*;  
 }  
  
 @Override  
 **public void** build(String airplaneType, **int** year, **int** engineCount){  
 **this**.**airplaneType** = airplaneType;  
 **this**.**year** = year;  
 **this**.**engineCount** = engineCount;  
 showInfo();  
 }  
  
 **public void** repair(){  
 System.***out***.printf(**"Repairing %s%n"**, ***AIRPLANE\_NAME***);  
 }  
  
 **public void** restore(){  
 System.***out***.printf(**"Restoring %s%n"**, ***AIRPLANE\_NAME***);  
 }  
  
 **public void** showInfo() {  
 System.***out***.printf(**"The %s Factory have built an airplane with following details:%n"**, ***AIRPLANE\_NAME***);  
 System.***out***.println(**super**.toString());  
 }  
}

**package** Question\_2;  
  
**public class** BertramFactory **extends** BoatFactory {  
 **static final** String ***BOAT\_NAME*** = **"Bertram Boat"**;  
 **private static** BertramFactory *bertramFactory*;  
  
 **private** BertramFactory() {  
 **super**();  
 }  
  
 **public static** BertramFactory getInstance() {  
 **if** (*bertramFactory* == **null**) {  
 *bertramFactory* = **new** BertramFactory();  
 } **else** {  
 System.***out***.printf(**"Warning! Only 1 %s Factory can be created. "** +  
 **"The existing factory will be used to build the new boat%n"**, ***BOAT\_NAME***);  
 }  
 **return** *bertramFactory*;  
 }  
  
 @Override  
 **public void** build(String boatType, **int** year, **int** length){  
 **this**.**boatType** = boatType;  
 **this**.**year** = year;  
 **this**.**length** = length;  
 showInfo();  
 }  
  
 **public void** showInfo() {  
 System.***out***.printf(**"The %s Factory have built a boat with following details:%n"**,***BOAT\_NAME***);  
 System.***out***.println(**super**.toString());  
 }  
  
 @Override  
 **public void** repair() {  
 System.***out***.printf(**"Repairing %s%n%n"**, ***BOAT\_NAME***);  
 }  
  
 @Override  
 **public void** restore() {  
 System.***out***.printf(**"Restoring %s%n%n"**, ***BOAT\_NAME***);  
 }  
}

**package** Question\_2;  
  
**public class** MastercraftFactory **extends** BoatFactory{  
 **static final** String ***BOAT\_NAME*** = **"Mastercraft Boat"**;  
 **private static** MastercraftFactory *mastercraftFactory*;  
  
 **private** MastercraftFactory() {  
 **super**();  
 }  
  
 **public static** MastercraftFactory getInstance() {  
 **if** (*mastercraftFactory* == **null**) {  
 *mastercraftFactory* = **new** MastercraftFactory();  
 } **else** {  
 System.***out***.printf(**"Warning! Only 1 %s Factory can be created. "** +  
 **"The existing factory will be used to build the new boat%n"**, ***BOAT\_NAME***);  
 }  
 **return** *mastercraftFactory*;  
 }  
  
 @Override  
 **public void** build(String boatType, **int** year, **int** length){  
 **this**.**boatType** = boatType;  
 **this**.**year** = year;  
 **this**.**length** = length;  
 showInfo();  
 }  
  
 **public void** showInfo() {  
 System.***out***.printf(**"The %s Factory have built a boat with following details:%n"**,***BOAT\_NAME***);  
 System.***out***.println(**super**.toString());  
 }  
  
 @Override  
 **public void** repair() {  
 System.***out***.printf(**"Repairing %s%n%n"**, ***BOAT\_NAME***);  
 }  
  
 @Override  
 **public void** restore() {  
 System.***out***.printf(**"Restoring %s%n%n"**, ***BOAT\_NAME***);  
 }  
}

**package** Question\_2;  
  
**public class** SeaRayFactory **extends** BoatFactory {  
 **static final** String ***BOAT\_NAME*** = **"Sea Ray Boat"**;  
 **private static** SeaRayFactory *seaRayFactory*;  
  
 **private** SeaRayFactory() {  
 **super**();  
 }  
  
 **public static** SeaRayFactory getInstance() {  
 **if** (*seaRayFactory* == **null**) {  
 *seaRayFactory* = **new** SeaRayFactory();  
 } **else** {  
 System.***out***.printf(**"Warning! Only 1 %s Factory can be created. "** +  
 **"The existing factory will be used to build the new boat%n"**, ***BOAT\_NAME***);  
 }  
 **return** *seaRayFactory*;  
 }  
  
 @Override  
 **public void** build(String boatType, **int** year, **int** length){  
 **this**.**boatType** = boatType;  
 **this**.**year** = year;  
 **this**.**length** = length;  
 showInfo();  
 }  
  
 **public void** showInfo() {  
 System.***out***.printf(**"The %s Factory have built a boat with following details:%n"**,***BOAT\_NAME***);  
 System.***out***.println(**super**.toString());  
 }  
  
 @Override  
 **public void** repair() {  
 System.***out***.printf(**"Repairing %s%n%n"**, ***BOAT\_NAME***);  
 }  
  
 @Override  
 **public void** restore() {  
 System.***out***.printf(**"Restoring %s%n%n"**, ***BOAT\_NAME***);  
 }  
}

1. Output

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. Main Source code

**import** Question\_1.PizzaDriver;  
**import** Question\_2.FactoryDriver;  
  
**public class** MainDriver {  
 **public static void** main(String[] args) {  
 PizzaDriver.*builderTest*();  
 FactoryDriver.*factoryTest*();  
 }  
}